

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 2003年 3月28日
Date of Application:

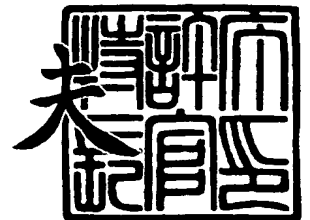
出願番号 特願2003-092381
Application Number:
[ST. 10/C]: [JP 2003-092381]

出願人 セイコーエプソン株式会社
Applicant(s):

2004年 1月30日

特許庁長官
Commissioner,
Japan Patent Office

今井 康夫



出証番号 出証特2004-3004749

【書類名】 特許願

【整理番号】 J0098698

【提出日】 平成15年 3月28日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 12/00

【発明者】

【住所又は居所】 長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社内

【氏名】 磯村 政一

【特許出願人】

【識別番号】 000002369

【氏名又は名称】 セイコーエプソン株式会社

【代理人】

【識別番号】 100066980

【弁理士】

【氏名又は名称】 森 哲也

【選任した代理人】

【識別番号】 100075579

【弁理士】

【氏名又は名称】 内藤 嘉昭

【選任した代理人】

【識別番号】 100103850

【弁理士】

【氏名又は名称】 崔 秀▲てつ▼

【手数料の表示】

【予納台帳番号】 001638

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0014966

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ベクトルデータのアドレス参照方法

【特許請求の範囲】

【請求項 1】 指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、

指標ベクトルの要素を格納する要素格納用レジスタを複数領域に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴とするベクトルデータのアドレス参照方法。

【請求項 2】 前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレスを算出することを特徴とする請求項 1 記載のベクトルデータのアドレス参照方法。

【請求項 3】 前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項 2 記載のベクトルデータのアドレス参照方法。

【請求項 4】 前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴とする請求項 1 記載のベクトルデータのアドレス参照方法。

【請求項 5】 前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項 4 記載のベクトルデータのアドレス参照方法。

【請求項 6】 前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴とする請求項 1～5 のいずれかに記載のベクトルデータのアドレス参照方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、ベクトルデータの読み出しあるいは書き込みにおけるアドレス参照方法に関する。

【0002】

【従来の技術】

従来、画像処理等において、不規則にメモリ上のデータの読み出しあるいは書き込むことが行われており、そのような処理を効率的に行うべく、種々の方法が提案されている。

例えば、画像処理において、1枚の画像データの中で、特定のブロックのデータをメモリからレジスタに格納する場合、ブロックの一行分の画素データをレジスタに格納した後、ブロックの次行の画素データを読み込むために、メモリ上の所定長離れたアドレスが参照される。

【0003】

このように、複雑なアドレス参照を行うための方法として、参考文献「スーパーコンピュータ」（オーム社、長島重夫、田中義一著）に記載された技術が知られている。

本文献においては、多次元配列データをメモリからベクトルレジスタに読み込んだり、ベクトルレジスタからメモリに書き込んだりする場合に、間接指標ベクトル参照を用いる技術が開示されている。

【0004】

間接指標ベクトル参照とは、メモリ上の参照するアドレス順序を格納したリスト（指標ベクトル）を用意しておき、そのリストを順に参照することによって、間接的にメモリ上の所定のアドレスを参照する方式である。

このような間接指標ベクトル参照を用いることにより、メモリ上の複雑なアドレス参照を行うことが可能となる。

【0005】

【非特許文献1】

長島重夫、田中義一著「スーパーコンピュータ」オーム社、p. 35
- 41

【0006】

【発明が解決しようとする課題】

しかしながら、従来の技術においては、指標ベクトルをベクトルレジスタに格納しておく必要があるため、以下のような問題が生じていた。

第1に、指標ベクトルを用意しておく手順が必要となるため、プログラムのコードサイズや処理サイクル数が増加してしまう。

【0007】

第2に、用意しておく指標ベクトルが増加すると、指標ベクトルを格納しておくベクトルレジスタも増加することから、本来の演算処理に用いるレジスタリソースが不足し、演算効率の低下を招いてしまう。

本発明の課題は、間接指標ベクトル参照をより効率的に行うことである。

【0008】

【課題を解決するための手段】

以上の課題を解決するため、本発明は、

指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、指標ベクトルの要素を格納する要素格納用レジスタ（例えば、図4のレジスタファイル40におけるインデックスベクトルを格納しているレジスタ）を複数領域（例えば、上位および下位の領域）に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴としている。

【0009】

例えば、発明の実施の形態における命令コード内の“エクステンション”によ

って、分割されたいずれかの領域を選択し、指定された領域のコードによって、指標ベクトル（インデックスベクトル）の要素を読み出すメモリ上のアドレスが指定される。そして、指定されたアドレスのデータを読み出していくことにより、指標ベクトルが生成される。

【0010】

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレス（例えば、ベースアドレス）に対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコード（例えば、上位3ビットあるいは下位3ビット等）と、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレス（メモリ上の参照アドレス）を算出することを特徴としている。

【0011】

この方法は、発明の実施の形態におけるインデックス修飾アドレッシングに関連する。

また、前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

【0012】

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、指標ベクトルの要素となるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴としている。

【0013】

この方法は、発明の実施の形態におけるポスト・レジスタ・アップデート・アドレッシングに関連する。

また、前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラレジスタに格納し、該スカラレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

【0014】

また、前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴としている。

本発明によれば、ベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割して、それぞれの領域に指標ベクトルの要素となる所定のコードを格納する。そして、分割した領域のいずれかを選択し、その領域に格納されたコードを用いて、所定の指標ベクトルを取得することとしている。

【0015】

したがって、1つの指標ベクトルを格納するレジスタの領域に、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

【0016】

【発明の実施の形態】

以下、図を参照して本発明に係るベクトルプロセッサの実施の形態を説明する。

本発明に係るベクトルプロセッサは、指標ベクトルを格納するベクトルレジスタにおいて、その要素レジスタを分割して使用し、アドレッシング機能を拡張している。

【0017】

したがって、初めに、このような機能を実現するための基本となる考え方について説明する。なお、本発明は、ベクトルプロセッサにおけるロード命令およびストア命令に関するものであるため、これらを中心に説明する。

ベクトルプロセッサにおいて、ロード命令あるいはストア命令のコードタイプには、アドレッシングモード（アドレス参照の方法）に応じて、以下の3種類が規定されている。

【0018】

図1は、ロード命令あるいはストア命令のフォーマットを示す図であり、(a)は、ベース相対アドレッシングに対応するLS0タイプ、(b)は、ポスト・

オフセット・アップデート・アドレッシングに対応するLS1タイプ、(c)は、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに対応するLS2タイプを示している。

【0019】

図1において、LS0タイプはスカラーデータのロード命令およびストア命令に対応しており、LS1タイプおよびLS2タイプは、スカラーデータおよびベクトルデータのロード命令およびストア命令に対応している。

本発明は、ベクトルデータのロード命令およびストア命令を取り扱うものであり、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに関するものであるため、LS2タイプについて説明する。

【0020】

図1(c)において、LS2タイプのフォーマットには、オペコード (opcode)、デスティネーション (dst)、ベース (base)、リピートアマウント (rptamt)、エクステンション (extension)、インデックス (index) の6つのフィールドが含まれている。

オペコードは、命令の内容を示すフィールドであり、ロード命令あるいはストア命令のいずれかを示すコードが含まれている。

【0021】

デスティネーションは、ロードあるいはストアの対象となるデータが格納されたレジスタのアドレスを指定するフィールドである。即ち、ロード命令の場合、メモリから読み出したデータを書き込むレジスタのアドレスであり、ストア命令の場合、メモリに書き込むデータを読み出すレジスタのアドレスである。

ベースは、基準となるメモリアドレスが格納されたレジスタのアドレスを指定するフィールドである。

【0022】

リピートアマウントは、ベクトル命令における命令の繰り返し回数 (要素データ数) を示すフィールドであり、命令繰り返し回数が“1”である場合はスカラー命令、“1”以外である場合はベクトル命令となる。

エクステンションは、機能拡張用に用意された5ビットのフィールドであり、

本発明においては、後述のように、エクステンションを利用して、分割した要素レジスタのアドレスを指定する。

【0023】

インデックスは、ベースに示されたアドレスに対し、どのような修飾を施すか、即ち、ベースに示されたアドレスから参照するアドレス順序を示すフィールドである。本発明においては、後述のように、インデックスを利用して、アドレス修飾の拡張仕様を規定する。

なお、インデックスは、上述の指標ベクトルが格納されたベクトルレジスタを指定しており、インデックスを参照することによって、間接指標ベクトル参照が行われる。

【0024】

続いて、本発明におけるアドレス修飾の拡張仕様について具体的に説明する。

図2は、インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図2においては、アドレス修飾の拡張仕様を説明するため、C言語に準じた記述スタイルを用いている。

【0025】

図2において、エクステンションが“000”である場合、インデックスとして指定されたレジスタの内容（指標ベクトル）を符号付きで解釈し、アドレス修飾を行う。

具体的には、ベースに示されたレジスタの値を常に基準とし、その値にインデックスに示されるアドレスを順次加算することにより、アドレス修飾を行う。

【0026】

また、エクステンションが“001”である場合、インデックスとして指定されたレジスタの下位16ビットを符号付きで解釈し、アドレス修飾を行う。

また、エクステンションが“010”である場合、インデックスとして指定されたレジスタの上位16ビットを符号付きで解釈し、アドレス修飾を行う。

さらに、エクステンションが“011”である場合、インデックスとして指定されたレジスタの下位16ビットおよび上位16ビットをそれぞれ符号付きで解

積し、交互にアドレス修飾を行う。具体的には、ベクトル命令における繰り返しの実行順序が偶数である場合、下位 16 ビットによってアドレス修飾を行い、奇数である場合、上位 16 ビットによってアドレス修飾を行う。

【0027】

次に、ポスト・レジスタ・アップデート・アドレッシングの拡張仕様について説明する。

図3は、ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図3においては、アドレス修飾の拡張仕様を実現するためのプログラミング例を併せて示している。

【0028】

図3において、エクステンションが“000”である場合、インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。

具体的には、ベースに指定されたレジスタの値を所定のレジスタ（図3中のパラメータTの値を格納するレジスタ）に一旦格納し、そのレジスタの値を更新しながらアドレス修飾を行う。

【0029】

また、エクステンションが“001”である場合、インデックスとして指定されたレジスタの下位 16 ビットを符号付きで解釈し、アドレスの更新を行う。

また、エクステンションが“010”である場合、インデックスとして指定されたレジスタの上位 16 ビットを符号付きで解釈し、アドレスの更新を行う。

さらに、エクステンションが“011”である場合、インデックスとして指定されたレジスタの下位 16 ビットおよび上位 16 ビットをそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。

【0030】

ここで、エクステンションが“000”～“011”までの場合、ベクトル命令における命令の繰り返し回数の間、インデックスとして指定された内容に応じてアドレス修飾が行われ、繰り返し回数が終了した後、ベースに指定されたレジスタの値が更新される。

したがって、複数のロード命令あるいはストア命令を用いて、繰り返しロードあるいはストアを行う場合、ベースに指定されたレジスタが自動的に更新されるので、命令発行の都度、ベースのアドレスを別の命令で更新する必要がなく、直ちに命令の実行を行うことが可能である。

【0031】

また、図3において、エクステンションが“100”～“111”の場合、エクステンションが“000”～“011”の場合とそれぞれ同様にアドレスの更新を行う。ただし、エクステンションが“100”～“111”の場合、命令の繰り返し回数が終了した後、ベースに指定されたレジスタの値を更新することなく命令の実行を終了する。

【0032】

次に、本発明に係るベクトルプロセッサの構成を説明する。

図4は、本発明を適用したベクトルプロセッサ1の構成を示す図である。

図4において、ベクトルプロセッサ1は、メモリ10と、メモリ制御部20と、命令フェッチ部30と、レジスタファイル40と、ロードユニット50と、ストアユニット60と、演算ユニット70とを含んで構成される。

【0033】

メモリ10は、ベクトルプロセッサ1に与えられる命令コードおよび演算対象となるデータを記憶している。

メモリ制御部20は、メモリ10に対するアクセス、即ち、データの読み出しや書き込みを制御する。例えば、メモリ制御部20は、ロードユニット50あるいはストアユニット60によって指定されたメモリ10のアドレスからデータを読み出したり、メモリ10から読み出されたデータをレジスタファイル40に出力したりする。

【0034】

命令フェッチ部30は、メモリ制御部20を介して、メモリ10から命令コードをフェッチし、一時的に記憶する。

レジスタファイル40は、32個のスカラレジスタSR0～SR31と、8個の要素レジスタからなる8個のベクトルレジスタVR0～VR7を含んで構成

され、メモリ 10 から読み出されたデータおよび演算結果を一時的に記憶する。

【0035】

ロードユニット 50 は、命令フェッチ部 30 に記憶された命令コードがロード命令である場合に、メモリ 10 から命令コードあるいはデータを読み出す処理を行う。

ストアユニット 60 は、命令フェッチ部 30 に記憶された命令コードがストア命令である場合に、メモリ 10 にデータを書き込む処理を行う。

【0036】

演算ユニット 70 は、命令フェッチ部 30 に記憶された命令コードが所定の演算命令である場合に、レジスタファイル 40 に記憶された所定データを対象として演算処理を行う。

ここで、ロードユニット 50 の構成について、詳細に説明する。

図 5 は、ロードユニット 50 の内部構成を示すブロック図である。

【0037】

図 5 において、ロードユニット 50 は、命令パイプライン制御部 51 と、インデックスレジスタ決定回路 52 と、デスティネーションレジスタ決定回路 53 と、アドレス演算回路 54 と、パイプラインレジスタ (PR) 55, 56 と、レジスタ 57 とを含んで構成される。

命令パイプライン制御部 51 は、ロードユニット 50 全体を制御するものである。

【0038】

インデックスレジスタ決定回路 52 は、命令コードのインデックスフィールドに基づいて、インデックスが格納されたインデックスレジスタを選択する信号 (インデックスレジスタ選択信号) を生成する。

なお、インデックス修飾アドレッシングの場合、基準として指定されたアドレス (ベースアドレス) を保持し、そのアドレスに対する各要素の相対アドレスを指定することで参照アドレスを生成する。即ち、インデックス修飾アドレッシングの場合、要素ごとに相対アドレスを指定するため、インデックスレジスタとしてベクトルレジスタが指定される。

【0039】

一方、ポスト・レジスタ・アップデート・アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を参照アドレスすると共に、そのアドレスに対する相対アドレスを指定してベースアドレスが更新される。そして、その更新されたベースアドレスに対する、次の要素の相対アドレスを指定することを繰り返して、参照アドレスを生成する。そのため、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタとベクトルレジスタの双方が指定可能である。

【0040】

デスティネーションレジスタ決定回路53は、命令コードのデスティネーションフィールドおよびリピートアmountフィールドに基づいて、デスティネーションアドレスを格納するためのデスティネーションレジスタを選択する信号（デスティネーションレジスタ選択信号）を生成する。

アドレス演算回路54は、命令パイプライン制御部51の指示に基づいて、レジスタファイル40から入力されるベースアドレスおよびインデックスアドレス（インデックスによって指定されるアドレス）に基づいて、ロード命令の対象となるメモリ10上のアドレス（ロードアドレス）を算出する。

【0041】

PR55, 56は、命令フェッチ部30から入力されるデスティネーションフィールドおよびリピートアmountフィールドのコードを一時的に記憶し、パイプライン処理において1サイクル遅延させて、デスティネーションレジスタ決定回路53に出力する。

レジスタ57は、命令フェッチ部30から入力されるベースフィールドのコードを一時的に記憶する。

【0042】

ここで、図5におけるアドレス演算回路54の構成について説明する。

図6は、アドレス演算回路54の構成例を示す図である。

図6において、アドレス演算回路54は、Iレジスタ54aと、Tレジスタ54bと、マルチプレクサ（MUX）54c～54eと、加算器54fとを含んで

構成される。

【0043】

Iレジスタ54aは、レジスタファイル40から入力されるインデックスアドレスを一時的に記憶する。

Tレジスタ54bは、MUX54cを介してレジスタファイル40から入力されるベースアドレスを一時的に記憶する。

MUX54cは、レジスタファイル40から入力されるベースアドレスと加算器54fの出力であるアップデート・ベースアドレスとのいずれかを切り替えて、Tレジスタ54bに出力する。

【0044】

MUX54dは、Iレジスタ54aから入力されるインデックスアドレスの上位あるいは下位のアドレスを選択して加算器54fに入力する。

MUX54eは、Tレジスタ54bから入力されるベースアドレスと加算器54fの出力であるアップデート・ベースアドレスとのいずれかを切り替えて、ベクトル命令の各繰り返しにおいて、ロード命令を実行するターゲットアドレスとして出力する。

【0045】

加算器54fは、Tレジスタ54bから入力されるベースアドレスの値と、MUX54dから入力されるインデックスアドレス上位あるいは下位のアドレスとを加算し、アップデート・ベースアドレスとして出力する。

なお、図6において、MUX54c～54eは命令パイプライン制御部51によって制御される。

【0046】

また、アドレス演算回路54は、インデックス修飾アドレッシングの場合およびポスト・アップデート・アドレッシングの場合それぞれに対応して、所定の動作を行う。

次に、動作を説明する。

初めに、図4を参照して、ベクトルプロセッサ1全体の動作について説明する。

。

【0047】

ベクトルプロセッサ1において処理が行われる場合、メモリ制御部20を介してメモリ10から命令フェッチ部30に命令コードが読み出される。

そして、ロードユニット50、ストアユニット60および演算ユニット70のそれぞれに、命令フェッチ部30から命令コードが出力される。

命令コードが入力されたロードユニット50、ストアユニット60および演算ユニット70は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【0048】

ここでは、図6を参照しつつ、命令コードがロード命令である場合について説明する。

命令コードのオペコードがロード命令を示している場合（より詳細にはprefixコードが“000”である場合）、ロードユニット50が動作する。

まず、ロードユニット50は、命令フェッチ部30から受け取ったベースフィールドのコードを、ベースレジスタ・リード選択信号（データの読み出し時にベースレジスタを選択する信号）としてレジスタファイル40に出力する。ベースレジスタ・リード選択信号は、レジスタファイル40内のスカラーレジスタSR0～SR31のいずれかをベースレジスタとして選択するための信号である。

【0049】

そして、ベースレジスタ・リード選択信号を受け取ったレジスタファイル40から、ベースレジスタ・リード選択信号によって指定されたレジスタに記憶されたベースアドレスの値がロードユニット50に入力される。

レジスタファイル40から入力されたベースアドレスの値は、アドレス演算回路54に入力される。

【0050】

また、インデックスレジスタ決定回路52は、命令フェッチ部30から入力されたインデックスフィールドのコードおよびリピートアマウントフィールドのコードを受け取る。そして、インデックスレジスタ決定回路52は、命令パイプライン制御部51からの指示に従って、命令コードがベクトル命令であるかスカラ

一命令であるかを判定し、ベクトル命令である場合、インデックスレジスタ選択信号を、リピートアマウントに示される要素データ数分、レジスタファイル40に順次出力する。このとき、インデックスレジスタ選択信号によって指定されたレジスタがベクトルレジスタである場合、指定されたベクトルレジスタ内の各要素レジスタを特定するための所定の選択信号が出力される。

【0051】

インデックスレジスタ選択信号が入力されたレジスタファイル40からは、インデックスレジスタ選択信号に示されるレジスタから、インデックスアドレスの値が、アドレス演算回路54に順次入力される。

インデックスアドレスの値を受け取ったアドレス演算回路54は、ベースアドレスの値と、インデックスアドレスの値とからロードアドレスを算出し、メモリ制御部20に出力する。なお、アドレス演算回路54の動作については後述する。

【0052】

また、命令フェッチ部30から入力されたデスティネーションフィールドのコードおよびリピートアマウントフィールドのコードは、パイプライン処理におけるタイミングを合わせるべく、PR55, 56に一旦記憶された後、デスティネーションレジスタ決定回路53に入力される。

すると、デスティネーションレジスタ決定回路53は、デスティネーションレジスタ選択信号を、メモリ10からロードされたデータと同期させてレジスタファイル40に出力する。

【0053】

すると、レジスタファイル40において、メモリ10からロードされたデータが、所定のデスティネーションレジスタに順次記憶される。

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、アドレス演算回路54によって出力されるアップデート・ベースアドレスをベースレジスタに書き込む。

【0054】

したがって、ベースフィールドのコードをレジスタ57に保持しておき、レジ

スタ 57 によって出力されるコードをベースレジスタ・ライト選択信号（データの書き込み時にベースレジスタを選択する信号）として用い、ベースレジスタ・ライト信号（ベースレジスタに対する書き込み指示信号）が入力されることに対応して、ベースレジスタのデータが更新される。

【0055】

次に、アドレス演算回路 54 の動作について説明する。

まず、インデックス修飾アドレッシングの場合について説明する。

インデックス修飾アドレッシングの場合、サイクル“1”において、ベースアドレスの値が、MUX 54 c を介して Tレジスタ 54 b に格納される。

一方、インデックスアドレスの値は、Iレジスタ 54 a に格納される。

【0056】

次いで、サイクル“2”において、Tレジスタ 54 b のベースアドレスの値と、Iレジスタ 54 a のインデックスアドレスの値とが、加算器 54 f によって加算され、MUX 54 e を介してターゲットアドレス（ロードアドレス）として出力される。

そして、第3サイクル以後は、ベースアドレスの値がTレジスタ 54 b に保持された状態のまま、新たにインデックスアドレスが入力され、加算器 54 f によって、ベースアドレスとの加算が順次行われる。

【0057】

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部 51 が、MUX 52 d を制御することにより、図 2 に示すインデックス修飾アドレッシングの拡張仕様が実現される。

即ち、エクステンションが“000”である場合、Iレジスタ 54 a に格納されたデータ（ここでは32ビット）が、そのまま加算器 54 f に入力される。

【0058】

一方、エクステンションが“001”である場合、Iレジスタ 54 a の下位 16 ビットを符号拡張し、32 ビットとされたデータが加算器 54 f に入力される。

また、エクステンションが“010”である場合、Iレジスタ 54 a の上位 1

6ビットを符号拡張し、32ビットとされたデータが加算器54fに入力される。

【0059】

さらに、エクステンションが“011”である場合、1サイクル毎に、Iレジスタ54aの下位16ビットと上位16ビットとを交互に選択し、32ビットに符号拡張されたデータが加算器54fに入力される。

続いて、ポスト・レジスタ・アップデート・アドレッシングの場合について説明する。

【0060】

ポスト・レジスタ・アップデート・アドレッシングの場合、サイクル“1”において、ベースアドレスの値が、MUX54cを介してTレジスタ54aに格納される。

一方、インデックスアドレスの値は、Iレジスタ54aに格納される。

次いで、サイクル“2”において、Tレジスタ54bのベースアドレスの値が、MUX54cを介して、そのままターゲットアドレス（ロードアドレス）として出力される。

【0061】

また、同時に、Tレジスタ54bのベースアドレスの値は、加算器54fにも出力され、Iレジスタ54aに記憶されているインデックスアドレスの値と加算される。

次いで、加算器54fの加算結果は、MUX54cを介して、Tレジスタ54bに格納され、Tレジスタ54bがベースアドレスとして記憶する値が更新される。

【0062】

そして、第3サイクル以後は、新たにインデックスアドレスが入力され、サイクル“1”，“2”と同様に、Tレジスタ54bの出力をターゲットアドレスとして出力すると共に、Tレジスタ54bがベースアドレスとして記憶する値が更新される。

その後、実行されているロード命令の最後の要素データのターゲットアドレス

を出力するのと同じタイミングで、加算器 54 f の出力がアップデート・ベースアドレスとしてレジスタファイル 40 に出力される。

【0063】

レジスタファイル 40 では、ベースレジスタ・ライト信号によって指示されたタイミングで、同時に入力されるベースレジスタ・ライト選択信号によって指定されたレジスタにアップデート・ベースアドレスを記憶する。

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部 51 が、MUX 52 d を制御することにより、図 3 に示すポスト・レジスタ・アップデート・アドレッシングの拡張仕様が実現される。

【0064】

即ち、エクステンションが“000”である場合、I レジスタ 54 a に格納されたデータ（ここでは 32 ビット）が、そのまま加算器 54 f に入力される。

一方、エクステンションが“001”である場合、I レジスタ 54 a の下位 16 ビットを符号拡張し、32 ビットとされたデータが加算器 54 f に入力される。

【0065】

また、エクステンションが“010”である場合、I レジスタ 54 a の上位 16 ビットを符号拡張し、32 ビットとされたデータが加算器 54 f に入力される。

さらに、エクステンションが“011”である場合、1 サイクル毎に、I レジスタ 54 a の下位 16 ビットと上位 16 ビットとを交互に選択し、32 ビットに符号拡張されたデータが加算器 54 f に入力される。

【0066】

また、エクステンションが“100”～“111”までである場合、エクステンションが“000”～“011”までの場合とそれぞれ同様の機能拡張を行うが、命令パイプライン制御部 51 がベースレジスタ・ライト信号を出力することなく、ベースレジスタの更新を行わないように制御する。

以上のように、本実施の形態に係るベクトルプロセッサ 1 は、インデックスによって指定されたベクトルレジスタの要素レジスタを複数の領域に分割し、分割

した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

【0067】

したがって、1つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

また、インデックスベクトルを用意する手順としては、1つのインデックスベクトル（指標ベクトル）の場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。

【0068】

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタが指定され、そのレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

そのため、インデックスレジスタとしてベクトルレジスタを用いる場合、ベクトルレジスタ全てにデータを格納した後でなければ後段の処理を開始できないのに対し、スカラーレジスタを使用できることから、スカラーレジスタにデータを格納した後、直ちに後段の処理を開始できる。

【0069】

したがって、インデックスベクトル（指標ベクトル）を用意する手順が軽減され、プログラムのコードサイズや処理サイクル数を減少させることが可能となる。

即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【0070】

なお、本実施の形態においては、命令コードのエクステンションフィールドを利用してアドレッシングの機能拡張を行うこととして説明したが、命令コードにアドレッシングの機能拡張のためのコードを含める場合の他、所定のレジスタを設け、そのレジスタに、アドレッシングの機能拡張のためのコードをセットする

こととしてもよい。

【0071】

また、本実施の形態においては、アドレッシングの機能拡張のために、レジスタを上位および下位の2つの領域に分割して使用することとして説明したが、3つ以上の領域に分けて使用することも可能である。

さらに、本実施の形態においては、命令コードに含まれるエクステンションによって、その命令全体における要素レジスタのアドレスを固定的に指定することとして説明したが、命令実行中に、要素レジスタの指定パターンを動的に変更することも可能である。

【図面の簡単な説明】

【図1】 ロード命令あるいはストア命令のフォーマットを示す図である。

【図2】 インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図3】 ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図4】 本発明を適用したベクトルプロセッサ1の構成を示す図である。

【図5】 ロードユニット50の内部構成を示すブロック図である。

【図6】 アドレス演算回路54の構成例を示す図である。

【符号の説明】

1 ベクトルプロセッサ、10 メモリ、20 メモリ制御部、30 命令フェッチ部、40 レジスタファイル、50 ロードユニット、51 命令パイプライン制御部、52 インデックスレジスタ決定回路、53 デスティネーションレジスタ決定回路、54 アドレス演算回路、54a Iレジスタ、54b Tレジスタ、54c～54e MUX（マルチプレクサ）、54f 加算器、55、56 PR（パイプラインレジスタ）、57 レジスタ、60 ストアユニット、70 演算ユニット

【書類名】

図面

【図 1】

(a) LS0タイプ(ベース相対アドレッシング)

opcode	dst	base	offset
--------	-----	------	--------

(b) LS1タイプ(ポスト・オフセット・アップデート・アドレッシング)

opcode	dst	base	rptamt	offset
--------	-----	------	--------	--------

(c) LS2タイプ(インデックス修飾／ポスト・レジスタ・アップデート・アドレッシング)

opcode	dst	base	rptamt	extension	index
--------	-----	------	--------	-----------	-------

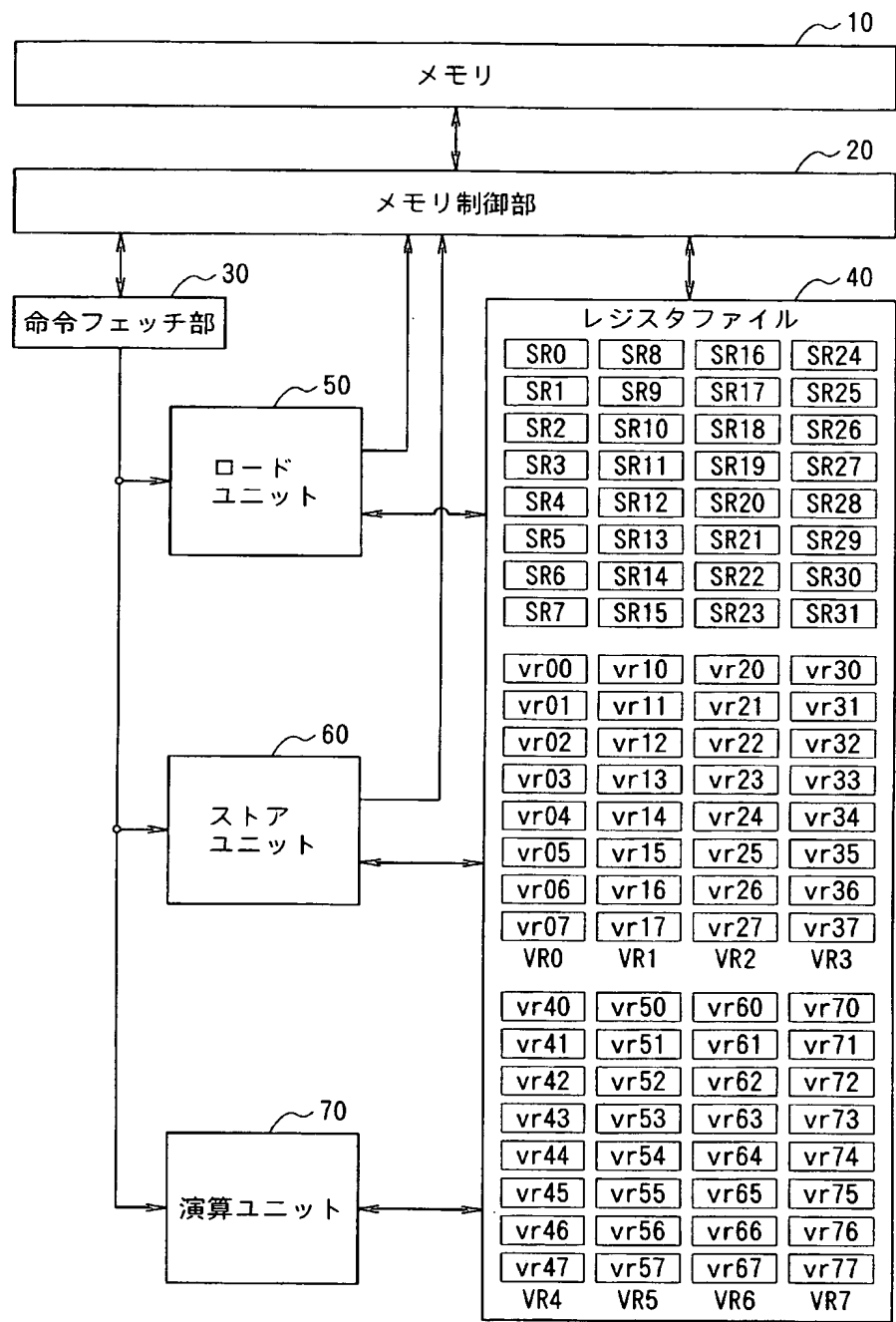
【図 2】

extension	アドレス修飾の方法
000	<p>インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[LOWER]; </pre>
001	<p>インデックスとして指定されたレジスタの下位 16ビット (index[t].h[LOWER]) を符号付きで解釈し、アドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[LOWER]; </pre>
010	<p>インデックスとして指定されたレジスタの上位 16ビット (index[t].h[UPPER]) を符号付きで解釈し、アドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[UPPER]; </pre>
011	<p>インデックスとして指定されたレジスタの下位 16ビット (index[t].h[LOWER]) および上位 16ビット (index[t].h[UPPER]) をそれぞれ符号付きで解釈し、交互にアドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) if ((t & 1) == 0) targetAdrs = base.uw + index[t].h[LOWER]; else targetAdrs = base.uw + index[t].h[UPPER]; </pre>

【図 3】

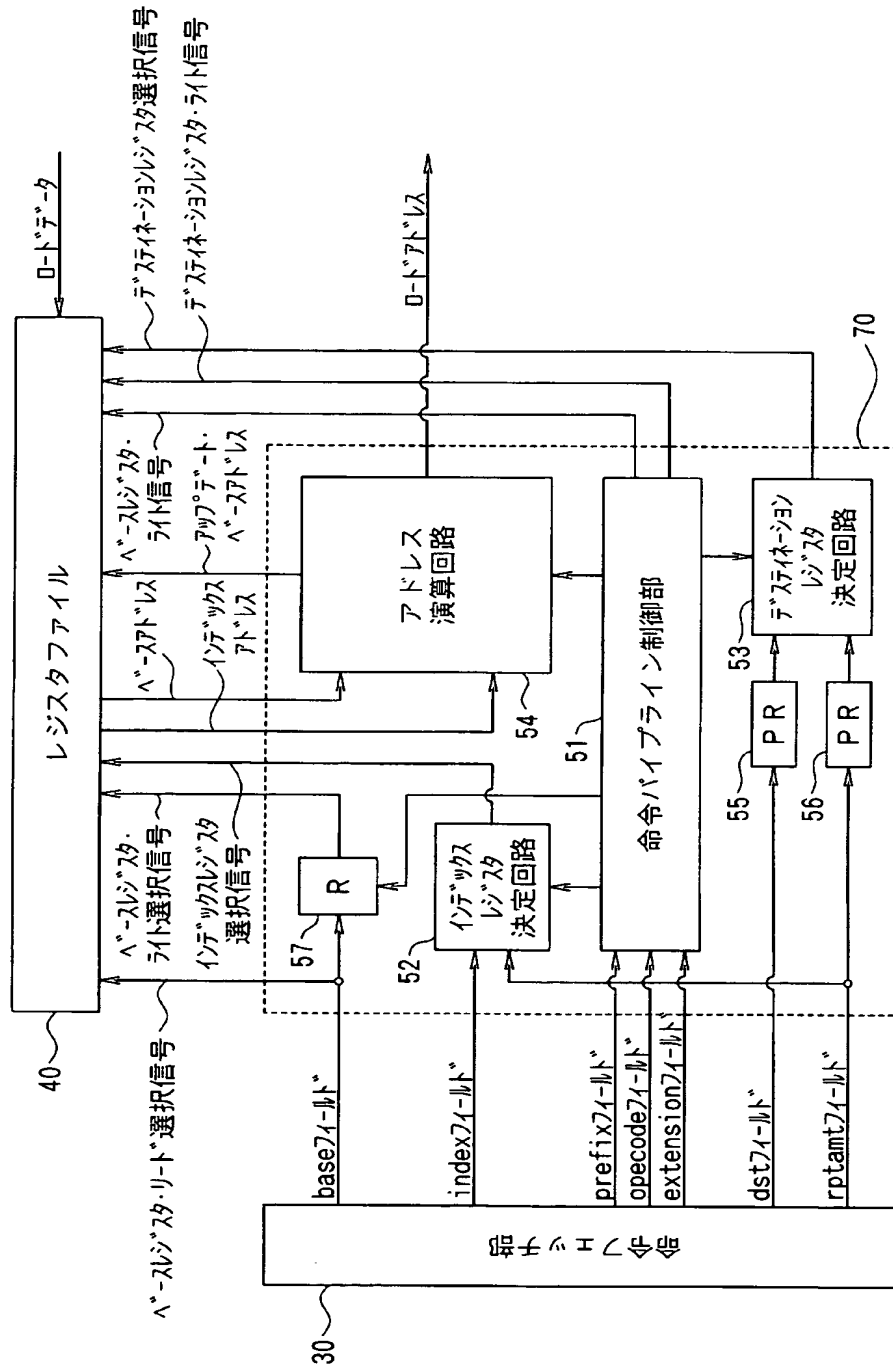
extension	アドレス修飾の方法
000	<p>インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].w; } base.uw = T; </pre>
001	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[LOWER]; } base.uw = T; </pre>
010	<p>インデックスとして指定されたレジスタの上位16ビットのindex[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[UPPER]; } base.uw = T; </pre>
011	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]をそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; if ((t & 1) == 0) T += index[t].h[LOWER]; else T += index[t].h[UPPER]; } base.uw = T; </pre>
100	<p>インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].w; } </pre>
101	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[LOWER]; } </pre>
110	<p>インデックスとして指定されたレジスタの上位16ビットのindex[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[UPPER]; } </pre>
111	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]をそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; if ((t & 1) == 0) T += index[t].h[LOWER]; else T += index[t].h[UPPER]; } </pre>

【図 4】

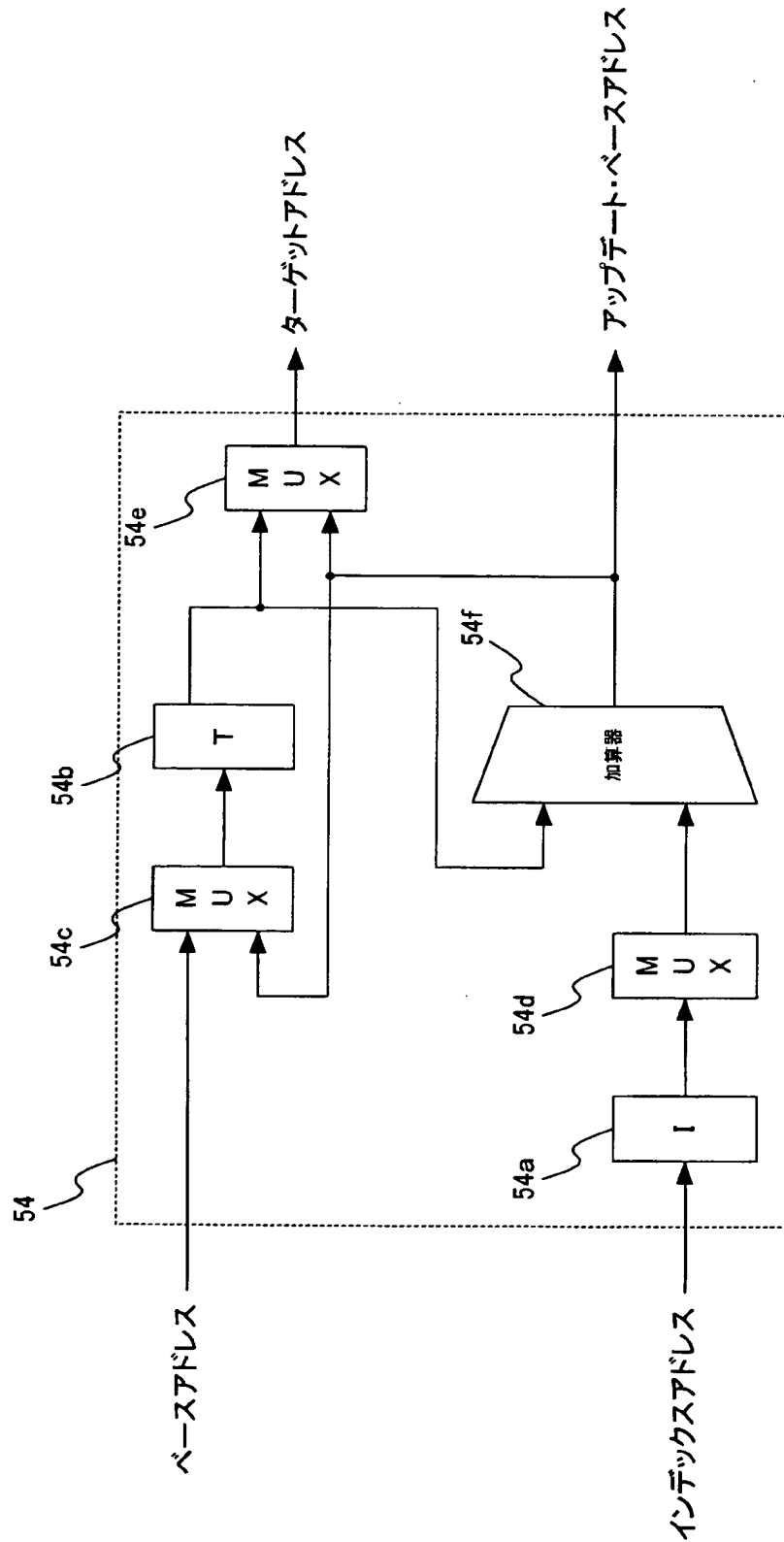


1

【図 5】



【図 6】



【書類名】 要約書

【要約】

【課題】 間接指標ベクトル参照をより効率的に行うこと。

【解決手段】 インデックスによって指定されたベクトルレジスタの要素レジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。したがって、1つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、ベクトルレジスタを効率的に使用することが可能となる。また、インデックスベクトルを用意する手順としては、1つのインデックスベクトルの場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【選択図】 図2

特願 2 0 0 3 - 0 9 2 3 8 1

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 2 3 6 9]

1. 変更年月日

1 9 9 0 年 8 月 2 0 日

[変更理由]

新規登録

住 所

東京都新宿区西新宿 2 丁目 4 番 1 号

氏 名

セイコーエプソン株式会社